**Version 2 tuning for computer vision problem.**

Objective.

To implement a content-based image retrieval system for indexed catalog of clothing patterns. This document outlines a deep learning powered approach based on feature vector comparison of clothing items.

Since this phase of the project is focused on upper body clothing item matching, the dataset of choice is the [deep fashion attribute prediction dataset](http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion/AttributePrediction.html) which consists of about ~70k labelled images of tops. The original dataset has been sorted to include only upper body wears and split into ~60k training and ~10k test data.

Architecture diagram.

A screenshot of a social media post

Description automatically generated

Fig 1.0

**Improvements over version 1.**

The goal is to get this model to at least 75% accuracy and use the last layers to extract the feature vector to be stored in our catalog. As far as improving the model goes here are the following approaches that have been tried.

* Merge training and validation dataset to increase number of training samples.
* Down sampling of train and test data set - Some classes had many times more images than others and the model bias to the majority class, trimming down classes 4,5 to have roughly $9k images might be useful here.
* Pre-process images by dividing pixel values by 255.0 since CNNs work better with smaller values between 0 - 1
* Fix issue with sparse accuracy method where string comparison instead of integer comparison prevented learning
* Reshuffle train and test data to have each batch represent the majority of the distribution.
* New data shape:

X\_train (39000, 32, 32, 3)

y\_train (39000,)

X\_test (5400, 32, 32, 3)

y\_test (5400,)

* Classification.

In order to create evenly distributed labels, some classes were merged with others and entire dataset was repicked. Here is the result of the repicking operation.

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Old label | New label | Total images |
| Anorak, bomber, jacket, parka | 1,4,11,13 | 0 | 6500 |
| Cardigan | 6 | 1 | 6500 |
| Sweater | 16 | 2 | 6500 |
| Tank | 17 | 3 | 6500 |
| Blouse | 3 | 4 | 6500 |
| Tee | 18 | 5 | 6500 |
| Total |  |  | **39000** |

**Model**

A screenshot of a social media post

Description automatically generated

Figure 2.0

[Model architecture used](Convolution%20with%2064%20different%20filters%20in%20size%20of%20(3x3)%20Max%20Pooling%20by%202%20ReLU%20activation%20function%20Batch%20Normalization%20Convolution%20with%20128%20different%20filters%20in%20size%20of%20(3x3)%20Max%20Pooling%20by%202%20ReLU%20activation%20function%20Batch%20Normalization%20Convolution%20with%20256%20different%20filters%20in%20size%20of%20(3x3)%20Max%20Pooling%20by%202%20ReLU%20activation%20function%20Batch%20Normalization%20Convolution%20with%20512%20different%20filters%20in%20size%20of%20(3x3)%20Max%20Pooling%20by%202%20ReLU%20activation%20function%20Batch%20Normalization%20Flattening%20the%203-D%20output%20of%20the%20last%20convolutional%20operations.%20Fully%20Connected%20Layer%20with%20128%20units%20Dropout%20Batch%20Normalization%20Fully%20Connected%20Layer%20with%20256%20units%20Dropout%20Batch%20Normalization%20Fully%20Connected%20Layer%20with%20512%20units%20Dropout%20Batch%20Normalization%20Fully%20Connected%20Layer%20with%201024%20units%20Dropout%20Batch%20Normalization%20Fully%20Connected%20Layer%20with%2010%20units%20(number%20of%20image%20classes)) is a popular one and used to classify cifarf-10 dataset. The entire model consists of 14 layers in total. In addition to layers below lists what techniques are applied to build the model.

Model is implemented using TensorFlow and the last layer is tweaked to classify images into 6 classes as opposed 10 classes for the original architecture.

* Results

The plot below shows the change in train and test accuracy with increasing epoch count. Test accuracy doesn’t seem to improve whereas train accuracy seems to fluctuate around the 0.4 mark.

|  |  |
| --- | --- |
| A screenshot of a map  Description automatically generated  Fig 3. Result from initial training  Best train accuracy: 0.426  Best test accuracy:0.26 | A close up of a map  Description automatically generated  Fig 4. Result after this phase of data tuning  Best train accuracy: 0.925  Best test accuracy:0.49 |

* Future plans

The goal is still to get this model to at least 75% accuracy on test data and use the feature layers to extract the feature vector to be stored in our catalog for comparison. As far as improving the model goes here are the following thoughts.

* Understand more about debugging convolutional neural networks and reducing overfitting.
* Experiment with a different neural network architecture known to perform nicely for this dataset.

(Started testing with residual neural network using keras api)

* Continue tweaking hyperparameters until better fit is found.